J. Fac. Cul. Edu. Saga Univ.
Vol. 3, No. 1 (1998) 29～44

アッカド語規則動詞活用表自動作成プログラム

塚　本　明　廣*

A program to generate paradigms of Akkadian strong verbs

Akihiro TSUKAMOTO

1．はじめに

紀元前3000年から紀元前後までの古代西アジアにおいては、東アジアにおける漢字文明圏に匹敵する一大文明圏が築かれていた。それは、当該文明を担った文字の名に因んで楔形文字文化圏とも称せるものであった。この文字によって記録された言語は、楔形文字の発明者と目される人々の使用言語であるシュメール語から、その継承者であり伝播者であったセム諸族の言語、さらにインド・ヨーロッパ語に属する古代ペルシャ語やヒッタイト語まで、実に多様である。当時の外交用語は、帝国の公用語であった、セム語族に属するアッカド語（二大方言であるバビロニア語とアッシリア語とを含む総称）であり、それが楔形文字を使用し続けたために、長期にわたり広範な影響力を与えたのである。ヒッタイト語や、半世紀ほど前に発見されたウガリット語、近年漸く姿を現したエブラ語など、楔形文字の解読によって初めてその全貌が知られ、あるいは今日知られ始めた言語や文化、そして民族は数多い（音声形式を言語分析のための最大の基盤とする）。現代の言語学者にとって幸いだったのは、漢字やエジプト聖刻文字と異なり、楔形文字が母音を表記した点にある。当時のメソポタミアに対峙する文明を誇ったエジプトも、その語彙の幾らかが同時代の楔形文字によって記録されたお陰で、僅かながらも、当時の母音体系の幾分かを推測できるのである[1)]。

さて、その楔形文字によって記録された主要言語の一つであるアッカド語は、解読から１世紀以上を経て、言語体系が大幅に解明された状態にあるとは言え、例えば動詞語幹の種類と機能に関しても、まだまだ決着をみない状況が随所に見られる。今日の標準的アッカド語文法を集大成したSodenがしばしば強調するように、個々の言語資料の精密な記述的研究を通して、これまでに推定された文法範疇の実証ないし検証を、今後一層推進する必要がある[2)]。

2．目的

本稿の目的は二つある。一つは主にアッカド語学習の観点から、酷似した紛らわしい形式が交錯して決

* 佐賀大学文化教育学部　日本・アジア文化講座

1) この項に関しては、主に次の文献が有益である。杉　勇『楔形文字入門』中公新書、1968。C. ウォーカー（大城光正訳、矢島文夫監修）『楔形文字』学芸書林、1995。亀井孝、河野六郎、千野栄一編『言語学大辞典』三省堂、1988の「アッカド語」（吉川守執筆）の項。エジプト語に関しては、V. デイヴィズ（矢島文夫監修、拙訳）『エジプト聖刻文字』学芸書林、1996。

2) Soden, §86c など。

して単純とは言えない動詞体系の習得のために、その学習過程でしばしば確認する必要に迫られる活用形を学習者に簡便な形で即座に提示することにある。さらに、一つは活用形の生成過程に関与する諸規則を言語学的観点から解明するために、コンピューターによるシミュレーションを通してその手掛かりを得ることにある。「手掛かり」とここで言うのは、言語学的に妥当と見なされる活用形生成の仕組みを、コンピューターを用いて文字どおり「機械的」に活用表を作成する仕組みに、必ずしも一対一に対応させ得ないことを認めるからである。また、場合によっては第一の目的を優先した結果、本稿ではその種の対応が徹底していない部分がある。さらに、本論文で扱うのは、強変化動詞、いわゆる「規則動詞」のみのパラダイムであり、しかも命令形も除外している。これらは、本研究の限界でもある。

3．先行研究と本研究の意義

コンピューターによる動詞生成のシミュレーションは既に実現されている可能性が考えられなくもないが、現在のところ一つとして公開されていない[3]。また、CD-ROM という記憶媒体があれば、すべての動詞についてその実証される活用形の一々を余すところなく入力することも可能であるから、動詞生成の仕組みをプログラム化する必要は、必ずしもない。インターネット上で公開されているアッカド語文法入門にしても[4]、活用表の一部を並べているにすぎない。したがって、本稿において動詞活用形が生成される仕組みをシミュレーションし、そのプログラムを公開し、批判と検討に晒すことは、無意味ではなかろう。本研究の意義は、その切っ掛けを提供することにある。

ここで試行された諸々の規則は、多くの研究者の地道な研究の積み重ねにより解明されてきたものである。膨大な粘土板文書に散在する個々バラバラの形式を体系化して、整然とした活用表に纏めあげるまでには、やはり数十年に渡る研究の蓄積が必要であった。Soden は、その集大成の一例である。

本研究の独創的な部分は、あくまでも、コンピューター・シミュレーションという手法によって、これまで個別的あるいは部分的に暗示され想定されてきた、動詞語幹とその活用形の生成の仕組みを、明示的に整合的に捉え直そうとした点にある。機械的処理あるいは SED を用いたシミュレーションの利点は、曖昧な処理を許さないことにある。そのため、場合によっては相互に矛盾する音韻規則の存在を浮かび上がらせることもある。それらを照らし合わせた上で、非文法的な形式を生成しないように厳密に規則の適用順序を調整しなければならない。本稿のこのような研究手法は、これまで音韻論の主流を占めてきた考え方に最も適合するものでもある。

いずれにしろ、本研究により、大した道具立ても必要とせず、ごく限られた記憶容量で、アッカド語動詞の基本的な活用形のほぼ全てを生成する仕組みを学習者に提供する可能性が示された[5]。

---

[3] 既にプログラミングを終えて本稿執筆中に、Reiner, E., *A Linguistic Analysis of Akkadian*, The Hague, 1966 を入手した。その補遺に当時の音韻論の手法による動詞生成規則が示されている。外見はプログラミングを思わせるものの、本稿の目的とする、ユーザーの要求に応じて活用形を表示する仕組みではない。とは言え、Reiner の生成規則は、動詞という抽象的品詞範疇がその直接構成素へ分肢・変換を重ねながら語幹を派生する過程を厳密に順を追って示した点において、参考にすべきことが多い。筆者は、Soden に対する見解の一部を第 4 回西アジア言語研究会(1998年 7 月 4 日、京都産業大学)において発表した。その際の質疑応答と Reiner とを含めて再検討したものを、稿を改めて年度内に他に発表する予定である。

[4] http://www.sron.ruu.nl/など。

[5] これを可能にしたのは、一群のフリー・ソフトウェアの存在である。本研究では特に GREP と SED に多大の恩恵を被っている。このような研究を可能にするツール類の存在があって初めて、本研究は着想され、実効性を試すことができた。このようなツール類を無償で提供された研究者に心からの謝意を表明したい。他に、バッチ・ファイルを実行する OS として、MS-DOS を使用した。

4．アッカド語動詞活用形構成素

4.1.動詞活用接辞

アッカド語動詞は人称（1・2・3）・数（単数・複数）・性（男性・女性）が融合した形式を動詞語幹に接辞して活用させる。その活用接辞には2種類ある。一つは現在形[6]・完了形・過去形に共通で、語幹に前接する（場合によってはさらに語幹に接尾辞が加わる形式を持つ）接辞体系（表1）と、一つはもっぱら状態形と命令形が関与する接尾形式のみからなる接辞体系（表2）とである。それらの接辞を備えた活用形の雛形を以下に示す。アッカド語文法では規則動詞の活用表を例示する場合に語根√prsを用いる習慣があるので、本稿もそれに倣う。ただし表1、表2では、接辞との違いを際立たせるために、語幹母音Vを含む語幹部分は大文字で表記した。小文字の部分が活用接辞である。各表の左端の略号は上で説明した3種の文法範疇の組み合わせで、例えば、m3sは男性・3人称・単数を意味する。#は語頭を、:は長音を示す。

表1

m3s	# iPRVS
f3s	# taPRVS
m2s	# taPRVS
f2s	# taPRVSi:
c1s	# aPRVS
m3p	# iPRVSu:
f3p	# iPRVSa:
m2p	# taPRVSa:
f2p	# taPRVSa:
c1p	# niPRVS

表2

m3s	# PRVS
f3s	# PRVSat
m2s	# PRVSa:ta
f2s	# PRVSa:ti
c1s	# PRVSa:ku
m3p	# PRVSu:
f3p	# PRVSa:
m2p	# PRVSa:tunu
f2p	# PRVSa:tina
c1p	# PRVSa:nu

4.2.動詞派生語幹形成接辞

アッカド語動詞には、主要語幹として4語幹、つまり基本語幹G（＝Grundstamm）と主に使役形を派生させる第2語根重複語幹D（＝Doppelungsstamm）、語幹形成素šを付加するŠ語幹、そして語幹形成素nを付加して主に受け身形を派生させるN語幹などが設定される。そして、その各々に対してさらに語幹派生の接中辞（-ta-，-tan-）が加わる表3のような動詞語幹体系が想定されている。この各語幹から、さらに現在形・完了形・過去形・命令形・不定詞・分詞・状態形などが形成されるのである（本稿は、そのうちの人称変化を行なう4活用形のみを考察する）。

表3

G	Gt		Gtn	
D	Dt		Dtn	
Š	Št1	Št2	Štn	ŠD
N	—		Ntn	
R	Rt			

[6] Sodenによる名称。HuehnergardやMiller & Shippではdurative。本稿では、durと略記した。

4.3.データ

活用表全体は、表3の語幹がそれぞれ現在形・完了形・過去形・状態形などの派生語幹を形成した上で、さらに表1または表2の接辞を組み合わせた形で構成される。ここではアッカド語文法の慣例に倣い、3人称・単数・男性形(m3s)を各語幹の代表形として掲げ、本稿の論述に必要な形式を一覧表示している(表4。ŠDの項の空欄は例証されないことを示す。極めて稀にしか例証されないRおよびRt語幹については、本稿では触れない)。

表4

語幹	現在形	完了形	過去形	状態形
G	iparras	iptaras	iprus	paris
Gt	iptarras	iptatras	iptaras	pitrus
Gtn	iptanarras	iptatarras	iptarras	pitarrus
D	uparras	uptarris	uparris	purrus
Dt	uptarras	uptatarris	uptarris	putarrus
Dtn	uptanarras	uptatarris	uptarris	putarrus
Š	ušapras	uštapris	ušapris	šuprus
Št1	uštapras	uštatapris	uštapris	šutaprus
Št2	uštaparras	uštatapris	uštapris	šutaprus
Štn	uštanapras	uštatapris	uštapris	šutaprus
ŠD	ušparras	uštaparris	ušparris	—
N	ipparras	ittapras	ipparis	naprus
Ntn	ittanapras	ittatapras	ittapras	itaprus

上述のとおり、実際には、表4に掲げた各形式が表1、表2の活用接辞に従ってさらに語形を変化させるのであるが、表4の派生形式間の関係が生成規則として抽出できるなら、人称接辞を伴うそれらの語形変化は人称等による形態の指定とすることで、言語学的にも無理なく説明でき、プログラムの面でも簡単に機械的に処理できる。したがって、本稿では表4の活用表の派生形式のみを取り上げる。

5．形態素配列規則

動詞活用形は、まず語根に派生語幹を形成する形態素が接辞して語幹を形成する段階がある。本節では、その際の個々の規則を帰納的に取り出す過程は省略し、試行錯誤によるシミュレーションを重ねて最終的に作成したバッチ・プログラムを、まず最初に(プログラムの実行には不要な行番号と各行毎の簡単な説明を付けて)示す。次に、処理の大まかな流れを解説する。

```
(1) ECHO OFF                          : 画面表示を消す
(2) IF "%1"==="" GOTO ?               : 使用法を表示するラベル?に飛ぶ
(3) IF "%2"=="stat" GOTO stat         : 状態形が入力されたらラベルstatに飛ぶ
(4) grep %3 base > bs                 : 他の活用形は表1の人称接辞を選択し、
(5) GOTO BS                           : 選択後、ラベルBSに飛ぶ
(6) :stat                             : ラベルstat
```

```
(7) grep %3 stat > bs                    :状態形は表2の人称接辞を選択し、次項へ。
(8) :BS                                  :ラベルBS
(9) IF "%1"=="g" GOTO G                  :入力された語幹に応じて、4ラベルに分岐
(10) IF "%1"=="gt" GOTO G                :同上
(11) IF "%1"=="gtn" GOTO G               :同上
(12) IF "%1"=="n" GOTO N                 :同上
(13) IF "%1"=="ntn" GOTO N               :同上
(14) IF "%1"=="d" GOTO D                 :同上
(15) IF "%1"=="dt" GOTO D                :同上
(16) IF "%1"=="dtn" GOTO D               :同上
(17) IF "%1"=="sd" GOTO D                :同上
(18) IF "%1"=="s" GOTO $                 :同上
(19) IF "%1"=="st" GOTO $                :同上
(20) IF "%1"=="sta" GOTO $               :同上
(21) IF "%1"=="stn" GOTO $               :同上
(22) :G                                  :ラベルG
(23) type bs > next                      :次のステップのためにファイル名を変更
(24) GOTO NEXT                           :ラベルNEXTに飛ぶ
(25) :N                                  :ラベルN
(26) sed s/P/NP/ bs > next               :第1語根の前に語幹形成素Nを挿入した
                                          後、
(27) GOTO NEXT                           :ラベルNEXTに飛ぶ
(28) :D                                  :ラベルD
(29) sed s/R/RR/ bs > next               :第2語根を重複した後、
(30) IF "%1"=="sd" GOTO sd               :語幹SDはラベルsdに飛ぶ
(31) GOTO D$                             :その他は、ラベルD$に飛ぶ
(32) :sd                                 :ラベルsd
(33) type next > bs                      :次の上書き処理のためにファイル名を変更
(34) :$                                  :ラベル$
(35) sed s/P/$P/ bs > next               :第1語根の前に語幹形成素$を挿入した後、
(36) :D$                                 :ラベルD$
(37) sed s/¥(#.¥)/¥1u/ next > D$         :語頭から2番目の位置に母音uを挿入し、
(38) type D$ > next                      :次のステップのためにファイル名を変更
(39) :NEXT                               :ラベルNEXT
(40) IF "%1"=="gt" GOTO ta               :語幹形成素 -ta- を持つ語幹はラベルtaに
                                          飛ぶ
(41) IF "%1"=="dt" GOTO ta               :同上
(42) IF "%1"=="st" GOTO ta               :同上
(43) IF "%1"=="sta" GOTO ta              :同上
(44) IF "%1"=="gtn" GOTO tan             :語幹形成素 -tan- を持つ語幹はラベルtan
```

```
                                                に飛ぶ
(45) IF "%1"=="ntn" GOTO tan                    : 同上
(46) IF "%1"=="dtn" GOTO tan                    : 同上
(47) IF "%1"=="stn" GOTO tan                    : 同上
(48) type next > nxt                            : 他は、次のステップのためにファイル名を
                                                  変更し、
(49) GOTO NXT                                   : ラベル NXT に飛ぶ
(50) :ta                                        : ラベル ta
(51) sed s/¥([P$]u*¥)/¥1ta/ next > nxt          : 第1語根か接辞 $ (と挿入母音 u) の後に ta
                                                  を挿入、
(52) GOTO NXT                                   : ラベル NXT に飛ぶ
(53) :tan                                       : ラベル tan
(54) IF "%2"=="dur" GOTO tan 2                  : 現在形ならラベル tan 2 に飛ぶ
(55) sed s/¥([P$N]u*¥)/¥1tan/ next > nxt        : 他は、語幹形成接辞 tan を指定位置に挿入
                                                  した後、
(56) GOTO NXT                                   : ラベル NXT に飛ぶ
(57) :tan 2                                     : ラベル tan 2
(58) sed s/¥([P$N]u*¥)/¥1tana/ next > nxt       : 語幹形成接辞 tana を指定位置に挿入後、
                                                  次項へ。
(59) :NXT                                       : ラベル NXT
(60) IF "%2"=="dur" GOTO DR                     : 現在形はラベル DR に飛ぶ
(61) IF "%2"=="pf" GOTO PF                      : 完了形はラベル PF に飛ぶ
(62) IF "%2"=="pret" GOTO PT                    : 過去形はラベル PT に飛ぶ
(63) IF "%2"=="stat" GOTO ST                    : 状態形はラベル ST に飛ぶ
(64) :DR                                        : ラベル DR
(65) IF "%1"=="g" GOTO DR 2                     : G 系列の語幹はラベル DR 2 に飛ぶ
(66) IF "%1"=="gt" GOTO DR 2                    : 同上
(67) IF "%1"=="gtn" GOTO DR 2                   : 同上
(68) IF "%1"=="n" GOTO DR 2                     : N 系列の語幹はラベル DR 2 に飛ぶ
(69) IF "%1"=="ntn" GOTO DR 2                   : 同上
(70) sed y/V/a/ nxt > last                      : 他の語幹は、語幹母音を a に書き換えた後、
(71) IF "%1"=="sta" GOTO RR                     : 語幹 Št2 はラベル RR に飛ぶ
(72) GOTO END                                   : 他は、ラベル END に飛ぶ
(73) :DR2                                       : ラベル DR 2
(74) sed -e y/V/%4/ -e y/o/a/ nxt > last        : 語幹母音を辞書の指定に従って書き込み、
(75) IF "%1"=="g" GOTO RR                       : 語幹 G はラベル RR に飛ぶ
(76) IF "%1"=="gt" GOTO RR                      : 語幹 Gt はラベル RR に飛ぶ
(77) IF "%1"=="gtn" GOTO RR                     : 語幹 Gtn はラベル RR に飛ぶ
(78) IF "%1"=="n" GOTO RR                       : 語幹 N はラベル RR に飛ぶ
(79) GOTO END                                   : 他は、ラベル END に飛ぶ
```

```
(80) :RR                                    : ラベル RR
(81) sed s/R/RR/ last > tmp                 : 第 2 語根を重複した後、
(82) type tmp > last                        : 次のステップのためにファイル名を変更し、
(83) GOTO END                               : ラベル END に飛ぶ
(84) :PF                                    : ラベル PF
(85) sed s/¥([PN$]¥)/¥1ta/ nxt > pf         : 指定位置に完了語幹形成接辞 -ta- を挿入
                                              し、
(86) IF "%1"=="g" GOTO PF2                  : G 系列の語幹はラベル PF 2 に飛ぶ
(87) IF "%1"=="gt" GOTO PF2                 : 同上
(88) IF "%1"=="gtn" GOTO PF2                : 同上
(89) IF "%1"=="n" GOTO PF2                  : N 系列の語幹はラベル DR 2 に飛ぶ
(90) IF "%1"=="ntn" GOTO PF2                : 同上
(91) sed y/V/i/ pf > last                   : 他は、語幹母音を i に書き換えた後、
(92) GOTO END                               : ラベル END に飛ぶ
(93) :PF2                                   : ラベル PF 2
(94) sed -e y/V/%4/ -e y/o/a/ pf > last     : 語幹母音を辞書の指定に従って書き込み、
(95) GOTO END                               : ラベル END に飛ぶ
(96) :PT                                    : ラベル PT
(97) IF "%1"=="g" GOTO PT2                  : G 系列の語幹はラベル PT 2 に飛ぶ
(98) IF "%1"=="gt" GOTO PT2                 : 同上
(99) IF "%1"=="gtn" GOTO PT2                : 同上
(100) IF "%1"=="n" GOTO PT2                 : N 系列の語幹はラベル PT 2 に飛ぶ
(101) IF "%1"=="ntn" GOTO PT2               : 同上
(102) sed y/V/i/ nxt > last                 : 他は、語幹母音を i に書き換えた後、
(103) GOTO END                              : ラベル END に飛ぶ
(104) :PT2                                  : ラベル PT 2
(105) sed y/V/%4/ nxt > pt                  : 語幹母音を辞書の指定に従って書き込んだ
                                              後、
(106) IF "%1"=="g" GOTO PT3                 : 語幹 G はラベル PT 3 に飛ぶ
(107) IF "%1"=="n" GOTO PT4                 : 語幹 N はラベル PT 4 に飛ぶ
(108) sed y/o/a/ pt > last                  : 母音交替語幹の母音を a に書き換えた後、
(109) GOTO END                              : ラベル END に飛ぶ
(110) :PT3                                  : ラベル PT 3
(111) sed y/o/u/ pt > last                  : 母音交替語幹の母音を u に書き換えた後、
(112) GOTO END                              : ラベル END に飛ぶ
(113) :PT4                                  : ラベル PT 4
(114) sed s/R[oa]/Ri/ pt > last             : 語幹母音を i に書き換えた後、
(115) GOTO END                              : ラベル END に飛ぶ
(116) :ST                                   : ラベル ST
(117) sed s/¥(#.¥)/¥1v/ nxt > st            : 語頭から 2 番目の位置に無指定母音を挿入
```

```
                                                    した後、
(118) IF "%1"=="g" GOTO ST2                         : 語幹Gはラベル ST 2に飛ぶ
(119) IF "%1"=="n" GOTO ST3                         : 語幹Nはラベル ST 3に飛ぶ
(120) IF "%1"=="gt" GOTO ST4                        : 語幹Gtはラベル ST 4に飛ぶ
(121) IF "%1"=="gtn" GOTO ST4                       : 語幹Gtnはラベル ST 4に飛ぶ
(122) IF "%1"=="ntn" GOTO ST5                       : 語幹Ntnはラベル ST 5に飛ぶ
(123) sed y/vV/uu/ st > last                        : 他は、挿入母音と語幹母音をu—uと書き
                                                      換え、
(124) GOTO END                                      : ラベルENDに飛ぶ
(125) :ST2                                          : ラベル ST 2
(126) sed -e y/vV/a%4/ -e y/o/i/ st > last          : 挿入母音aと辞書指定の語幹母音かiに書
                                                      き換え、
(127) GOTO END                                      : ラベルENDに飛ぶ
(128) :ST3                                          : ラベル ST 3
(129) sed y/vV/au/ st > last                        : 挿入母音と語幹母音をa—uと書き換え、
(130) GOTO END                                      : ラベルENDに飛ぶ
(131) :ST4                                          : ラベル ST 4
(132) sed y/vV/iu/ st > last                        : 挿入母音と語幹母音をi—uと書き換え、
(133) GOTO END                                      : ラベルENDに飛ぶ
(134) :ST5                                          : ラベル ST 5
(135) sed -e s/Nv/i/ -e y/V/u/ st > last            : 接辞削除、挿入母音と語幹母音をi—uと書
                                                      き換え、
(136) GOTO END                                      : ラベルENDに飛ぶ
(137) :?                                            : 使用法表示のためのラベル
(138) ECHO Input: %%1 :g/gt/gtn/d/dt/dtn/s/st/sta/stn/sd/n/ntn
                                                    : ①語幹名の入力を促す表示
(139) ECHO Input: %%2 :dur/pf/pret/stat             : ②活用形の入力を促す表示
(140) ECHO Input: %%3 :mfc/321/sp                   : ③人称接辞の入力を促す表示
(141) ECHO Input: %%4 :o/a/i/u                      : ④語幹母音の入力を促す表示
(142) :END                                          : ラベル END
(143) sed -gf final.sed last                        : SED スクリプトの実行（次節参照）
```

5.1.解説 （行頭の数字は上記バッチ・ファイルの行番号である）

137-141：語幹毎の活用表を出力するのに必要な4項目を指定（入力）するよう、ユーザーに促す。4項目とは、①表3の語幹のいずれか(ただし、gotoコマンドで使用するラベルとの関係で、St̆1はstに、St̆2はstaとしている)、②現在形（dur）・完了形（pf）・過去形（pret）・状態形（stat）のいずれか、③男性形（m）・女性形（f）・共通形（c）のいずれか、④語根毎に語彙的に、つまり辞書において指定される語幹母音（ここでoとしたのは、いわゆる母音交替（Ablaut）語幹といわれるもので、a / i / uのいずれでもない母音というつもりである)。

3-7：ユーザーの指定（入力②）に従い2種の活用接辞のいずれかを自動的に選択し（入力がstatの場

合は表2を、それ以外は表1を選択。これらは、statとbaseという二つのファイルに格納されている。)、その一覧表の中からさらに、人称・数・性に応じた活用接辞を伴った語幹を選択する。ここで基本形（bs）が出力される。

8-21：ユーザーが指定(入力①)した語幹に飛ぶ。語幹に加わる接辞により、G、N、D、Šの4系列が認められる。

22-24：G系列（G、Gt、Gtn）は、特に接辞を加えることがない。

25-27：N系列（N、Ntn）は、語幹の第1子音の前に語幹形成接辞Nを付加する。Ntは存在しない。

28-29、31、36-39：D系列(D、Dt、Dtn)は、29で語幹第2子音を重複させた後、さらに37で語頭から2番目の位置に母音uを挿入する。ここで、接頭型接辞（表1）の場合に非文法的な母音連続が生じるが、それは、次節（§6）で述べる音韻規則9で削除して調整される。

34-39：Š系列(Š、Št1、Št2、Štn、ŠD)は、35で語幹第1子音の前に語幹形成接辞Šを付加した後、37で語頭から2番目の位置に母音uを挿入する。母音連続に関しては、D系列の場合と同じ処理が施される。

28-30、32-39：Š系列の内、ŠDのみは29でD系列の、35と37でŠ系列の、つまり両方の語幹形成接辞を適用される。

40-49：以上4系列毎の語幹形成を終えて出力された形式（next）を、接中辞 -ta- または -tan- を挿入する規則を適用するための入力形として、振り分ける。

50-52：接中辞 -ta- は、第1語根Pまたは語幹形成接辞Šの後に、ただしD系列語幹およびŠ系列語幹の状態形では語幹形成の際に挿入された母音uの後に、挿入される。Ntは存在しない。

53-58：接中辞 -tan- は、現在形では -tana- の形、その他の活用形では -tan- の形をとり、いずれも第1語根Pまたは語幹形成接辞NまたはŠの後に挿入する。状態形における挿入母音uの扱いは、接中辞 -ta- の場合と同じである。

60-63：以上の語幹拡張規則の適用を受けた出力形（nxt）を、現在形・完了形・過去形・状態形の4形式の入力形として振り分ける。

64-69、71：現在形は、G、N両系列とD、Š両系列で派生の方法が大きく異なる。また、Št2も一部特異であるため、振り分ける必要がある。

70：D系列およびŠt2を含む系列では、語幹母音をaに変えて現在形を生成する。

73-83：G、N両系列の現在形は、辞書で指定された母音を維持するが、その内のNtn以外の語幹とŠ系列中のŠt2語幹においては、第2語根を重複させることにより他の時制との弁別を行う。

84-85、91-92：完了形は、全系列において第1語根または語幹形成接辞N、Šの後に接中辞 -ta- を挿入する。その際、D、Š両系列においては同時に語幹母音をiに変える。

86-90、93-95：G、N両系列の完了形では、語幹母音は辞書の指定を維持する。

96 、102-103：過去形では、G、N以外の系列つまりD、Š両系列において語幹母音をiに変える。

97-101、104-115：G、N両系列においては、過去形形成の際の語幹母音の振る舞いが様々である。GおよびNを除く語幹の場合、語幹母音は辞書の指定を維持し、母音交替語幹では母音はaである。しかし、GおよびNに関しては、独自の規則が必要である。Gでは母音交替語幹で語幹母音がuとなり(111)、Nでは母音交替語幹とおよび指定母音aを持つ語幹とで母音がiとなる（114)。

116-136：状態形については、語頭から2番目の位置に挿入される母音（v）と語幹母音（V）との組み合わせに5つのパタンが見られる（下の表5を参照。ただし、D系列およびŠ系列に関しては母音の型が同じなので、代表語幹しか示していない。)。したがって、規則にもそれを反映させている。もっとも、規則

132と135とは、語幹形成素 n の関与を除けば母音の組み合わせの型そのものは同じである[7]。

表5

語幹母音：	母音交替	a	i	u	適用規則
D	purrus	同左	同左	同左	123
Š	šuprus	同左	同左	同左	123
G	paris	labaš	paqid	—	126
N	naprus	同左	paqud（同左）	namgur（同左）	129
Gt	pitrus	同左	mitluk（同左）	ritgum（同左）	132
Gtn	pitarrus	同左	mitalluk（同左）	ritaggum（同左）	132
Ntn	itaprus	同左	itapqud（同左）	—	135

5.2.考察

以上の語幹形成規則は、辞書的意義を担う語根に、文法的意義を担う諸々の形態素が関与して出力される形式を扱ったものである。これらの規則は、まず第一に、形態素の挿入位置を指定したり交替すべき形態素を指定する点に特徴がある。第二に、共通の形態論的特徴（子音重複や母音交替の型を含む）を持つ語幹同士を振り分けるという特徴があった。この振り分けの過程で、母音の型に関して、D 系列と Š 系列に、また G 系列と N 系列に共通点があること、さらに、語幹 G と語幹 N に現在形の子音重複などの共通性が見られること、などが確認された。

一方、さらに考察・検討すべき点も明らかになった。例えば接中辞 -tan- は、ほとんどの語幹において現在形にしか明白な形で現れない。現在形以外ではほぼ全ての形式で -ta- 拡張語幹と同形意義をなしている。同形意義は、この他にも、G の完了形と Gt の過去形、Gt の現在形と Gtn の過去形、D・Š・N の各完了形と Dt(n)・Št(1,2,n)・Ntn の各過去形に見られ、アッカド語動詞体系は同形意義で溢れている。文法体系として不都合はなかったのか、言い換えれば、動詞体系は正しく再建されたのであろうか。

これに対する言語学的解釈はひとまず別の機会に譲り、こうして形成された形態素結合のみからなる語幹から、音節構成等、語の表面形を調整して文法的形式を出力するためには、さらに次節の音韻規則を適用する必要がある。

6．音韻規則

本節では、前節で扱った形態素結合規則を適用された結果出力された形式（last）が、アッカド語の単語として認定されるのに必要な形式つまり文法的形式へと、表面形を整える規則を扱う。前節同様、まず規則を掲げた後に解説する。ここで適用される規則のほとんどは、原則的に形態論的条件が付かない。また、音韻的条件さえ整えば、動詞に限定されることなく、他の品詞にも適用される性質を持つことが予想される規則である。この一連の音韻規則は、SED スクリプト・ファイルとしてファイル名 final に収められ、上記バッチ・ファイルの最後の行（143）で実行される。

---

[7] これを骨格、子音層、母音層からなる多重構造と捉え直す最近の理論的背景（自律分節音韻論）については、ごく簡略ながら、Katamba,F., An Introduction to Phonology, London, 1989 を参照されたい。

(1) #e
(2) #def c 'bdgxyklmnpqrsS$tTwzPRS
(3) #def v aiuV
(4) s/n¥(@c@c¥)/¥1/
(5) s/[Nn]¥(@c¥)/¥1¥1/
(6) s/¥(@c¥)¥1¥(@c¥)/¥1¥1a¥2/
(7) s/¥(@c@c*¥)¥(@c@c¥)/¥1a¥2/
(8) s/¥(@v@c¥)@v¥(@c@v¥)/¥1¥2/
(9) s/u@v/u/

6.1.解説

1：定義宣言行
2：子音定義行。アッカド語の子音音素目録、大文字表記した語根子音を含む。
3：母音定義行。アッカド語の母音音素目録、大文字表記した語幹母音を含む。
4：語幹形成素N以外のnを頭音とする3子音連続があれば、頭音のnを削除する[8]。
例えば、DtnやNtnの過去形などで： up-tan-rris > uptarris, in-tan-pras > ittapras
5：語幹形成素Nを含むnを頭音とする2子音連続があれば、nが後続子音に同化する。
例えば、Gtnの過去形やNtnの現在形などで： ip-tan-ras > iptarras, in-parras > ipparras
6：重複子音を頭音とする3子音連続があれば、重複子音の後に母音aを挿入する。
例えば、Nの過去形などで： i-n-pris > ippris (5の適用)> ipparis
7：3子音または4子音連続があれば、母音aを挿入して子音連続を分割する。
例えば、Š、ŠDの現在形などで： u-š-pras > ušapras, u-š-p-rras > ušparras
8：開音節が2つ連続する場合、2番目の開音節の母音を削除する[9]。
例えば、Gtの完了形や状態形などで： ip-ta-ta-ras > iptatras, pi-ta-rus > pitrus
9：母音uの後に他の母音が続く場合、uを残して他を削除する（uは他の母音を上書きする）。
例えば、DやDtの現在形などで： u-iparras > uparras, u-ip-ta-rras > uptarras

6.2.考察

これらの音韻規則に関して、音韻理論上最も問題となるのは、規則の適用順序である。果たして、音韻規則には適用すべき厳格な順序が定められていて、しかもそれぞれの規則はただ一度しか適用されないのか。それとも、適用順序など存在せず、音韻条件さえ満たせば、場合によっては何度でも、適用されるのか。本研究で使用したSEDは、スクリプトに並べられた書き換え命令を上から順に実行する仕組みを備えているため、適用順序の検証には格好のプログラムである。以下、主として適用順序の問題を検討する。音韻規則4と5から判断する限り、適用順序の指定は必要である。規則5の方が先に適用されると、その後の規則の適用によって、例えば過去形において次のような非文法形式（*を付けた形式）が派生する：

N	Dtn	Ntn	適用規則

[8] Soden、§20i。
[9] Soden、§12b、c。

i-N-pris	u-p-tan-rras	i-n-tan-pras	
*ippris	*uptarrras	*ittappras	5 ([Nn]C > CC)
—	—	—	4 (nCC > CC) [適用不能]
ipparis	*uptarraras	*ittapparas	6 (C_1C_1C > C_1C_1aC)

いずれも、規則5の適用によって規則4の適用条件が消滅し、それが適用されなかった結果生じた形である（ただし、Nの過去形については問題ない）。たとえ規則4の適用条件を「3子音の連続」と緩和しても、非文法的形式の生成は避けられない。ただし、非文法的形式を生じるのは、今度はNの過去形の方である：

N	Dtn	Ntn	適用規則
i-N-pris	u-p-tan-rras	i-n-tan-pras	
*ippris	*uptarrras	*ittappras	5 ([Nn]C > CC)
*ipris	uptarras	ittapras	4 (C_1CC > CC)
—	—	—	6 (C_1C_1C > C_1C_1aC) [適用不能]

したがって、以下に示すとおり、規則4は規則5の実行前に適用しなければならない：

N	Dtn	Ntn	適用規則
i-N-pris	u-p-tan-rras	i-n-tan-pras	
*i-N-pris	up-tarras	*in-tapras	4 (nCC > CC)
*ippris	—	ittapras	5 ([Nn]C > CC)
ipparis	—	—	6 (C_1C_1C > C_1C_1aC)

ただし、途中で非文法的形式（*ippris）を生じる点は、認知音韻論で問題となるかもしれない[10]。

ところで、規則6の適用条件を備えた形式は規則5が適用されて初めて生じる形式である。したがって、音韻規則が1回限りしか適用できない性質を持つものであるなら、5と6の適用順序を変更することは必然的に規則6を適用しないこと、つまり規則6は存在しないことを意味し、非文法的形式を生じる：

N	Dtn	Ntn	適用規則
i-N-pris	u-p-tan-rras	i-n-tan-pras	
*i-N-pris	up-tarras	*in-tapras	4 (nCC > CC)
—	—	—	6 (C_1C_1C > C_1C_1aC) [適用不能]
*ippris	—	ittapras	5 ([Nn]C > CC)

規則6と規則7の場合も、次の例のように適用順序を変えることができない：

[10] 認知音韻論については、Goldsmith, J. (ed.), *The Last Phonological Rule*, Chicago, 1993。特にその中のLakoff, G., Cognitive Phonologyには多くの示唆を得たものの、本稿ではその構想をほとんど実現できなかった。一因は、その目的に応じたツールが手許にないためであるが、あるいは、筆者がSEDの仕様を充分に生かし切っていないのかもしれない。御教示願いたいところである。

N の過去形	Ntn の完了形	Ntn の過去形	ŠD の現在形	適用規則
i-N-pris	i-N-ta-tan-pras	i-N-tan-pras	u-š-pras	
*ippris	ittatapras	ittapras	—	4 & 5
*ipapris	—	—	ušapras	7 (CCC > CaCC)
—	—	—	—	6 (C_1C_1C > C_1C_1aC) [適用不能]

これに対し、適用順序を変えた次の例ではいずれも文法的形式を生成している：

N の過去形	Ntn の完了形	Ntn の過去形	ŠD の現在形	適用規則
i-N-pris	i-N-ta-tan-pras	i-N-tan-pras	u-š-pras	
*ippris	ittatapras	ittapras	—	4 & 5
ipparis	—	—	—	6 (C_1C_1C > C_1C_1aC)
—	—	—	ušapras	7 (CCC > CaCC)

以上のような結果に基づき、本節に掲げた規則は、上の順序で一回限り適用される場合を想定した配列になっている。それは、基本的には、規則の適用順序を外的に指定するという、これまでの音韻論で有力であった立場を反映したためであるが、他方、シミュレーションに使用したSEDというプログラムの仕様から必然的にもたらされた制約でもある。シミュレーションは別としても、少なくとも、適用順序の指定に反論できるような解決法は、今のところ見出していない。

ところが一方、ある形式が適用条件を備えさえすればいつでも、何度でも規則が適用され効力を発揮すると仮定した場合は、規則6と規則5の適用順序はどちらが先でも出力結果は異ならない：

N	Dtn	Ntn	適用規則
i-N-pris	u-p-tan-rras	i-n-tan-pras	
i-N-pris	up-tarras	in-tapras	4 (nCC > CC)
—	—	—	6 (C_1C_1C > C_1C_1aC) [適用不能]
*ippris	uptarras	ittapras	5 ([Nn]C > CC)
ipparis	—	—	6 (C_1C_1C > C_1C_1aC)

この場合は、結局、規則の適用順序が固定されていないのと同じ効果をもたらしている。生成過程で一旦生じた非文法的形式が規則6の適用条件を備えたために、直ちにその適用を受け、結果的に文法的形式に変換されたのである。もっとも、上の例示から明らかなように、規則6は単に規則5の前で適用されていないだけだとも言える。これと同様の状況は、規則7と規則8との関係にも見られる。この二つの規則は、一方が母音を挿入し、他方が母音を削除するという相反する効果を発揮するものである。つぎの派生過程を参照されたい（ŠDの現在形では規則7が2回適用されているとも考えられる）：

Gt の完了形	Gt の状態形	Šの過去形	ŠD の現在形	適用規則
ip-ta-ta-ras	pi-ta-rus	u-š-pris	u-š-p-rras	

ipta-t-ras	pi-t-rus	—	—	8 (CVCVCV > CVCCV)
—	—	uš-a-pris	*ušaparras	7 (CCC > CaCC)
—	—	—	ušparras	8 (CVCVCV > CVCCV)

上の出力結果は、規則7と8をその順序で（ただし、規則7を2回）適用した下の場合と異ならない：

Gt の完了形	Gt の状態形	Šの過去形	ŠD の現在形	適用規則
ip-ta-ta-ras	pi-ta-rus	u-š-pris	u-š-p-rras	
—	—	uš-a-pris	*ušaparras	7 (CCC > CaCC)
ipta-t-ras	pi-t-rus	—	ušparras	8 (CVCVCV > CVCCV)

もっとも、適用回数に制限がなくとも、規則4と5の場合には、順序が変われば非文法的な形式が生じる：

N	Dtn	Ntn	適用規則
i-N-pris	u-p-tan-rras	i-n-tan-pras	
*ippris	*uptarrras	ittappras	5 ([Nn]C > CC)
—	—	—	4 (nCC > CC) [適用不能]
—	—	—	5 ([Nn]C > CC) [適用不能]
ipparis	*uptarraras	*ittapparas	6 ($C_1C_1C > C_1C_1aC$)

このように、一部には適用順序を定めないと非文法的形式を生み出してしまう規則があると同時に、一部には適用順序に関係なく文法的形式を生み出す規則があることもまた、事実である。さらに、最終的には適用順序に影響されない規則の中にも、一旦は他の規則の適用を受けて非文法的形式を生じてしまう場合もあれば、派生途上でも他の規則の適用環境に影響を受けることも与えることもない規則もある。規則9は、その種の、他の規則との交渉がない例である。そのような規則間の無干渉、無交渉とも言える状況は、SEDの仕様のままでも、該当する規則の位置を交換したり、規則1から9に至る一連の規則全体を再適用することで確認できる。これは、上で検討したとおりである。生成途上で生じる非文法的形式を出力して確認することも可能である。

生成過程で非文法的形式を生じても、生成途上の形式は表面に現れることは決してないため、本稿では、最終的に文法的形式を整えていれば特に問題としなかった。そのことは、規則の適用順序を指定することと共に、生成音韻論に対して認知音韻論が鋭い批判を向ける点であるが[11]、ここでは言及するに止める。

7．結論および今後の課題

以上のシミュレーションの結果、今後解明を要することがらは少なくないことが明らかになった。例えば語幹形成に関して言えば、語幹拡張形態素 -tan- の現われ方が現在形でのみ特殊なこと、現在形語幹形成のための形態論的手段が単一でないこと、一面でそのように多様な語幹形成手段が利用されるにも拘わらず同形意義が極めて多いこと、などである。

[11] 前註を見よ。

同様に、音韻規則に関しても、今後さらに検討すべきことがらが多い。例えば、規則7で3子音連続と4子音連続を同一規則で処理するのは、適切か。ŠDの現在形で検討したように（§6.2.）、4子音連続の場合は、母音挿入によって3子音連続の音節構成を調整する過程が2回関与しているのではないのか。つぎに、nの同化あるいは削除が関与する二つの規則（4と5）において、語幹構成素のNと派生語幹（-tan-）の一部に含まれるnとが異なる振る舞いをするのは、規則の立て方に問題があるのではないか。さらに、音韻規則の3つの規則（4、6、7）は、3子音以上の子音連続を解消してCVまたはCVCの音節からなる連鎖へと単語の音節構成を整える規則である。同一の機能を持つ規則が3種類も存在し得るのか。特に、Sodenの想定する規則4は果たして音韻論的に妥当なものであろうか。最後に、認知音韻論では規則の適用順序を指定することに反対する。それを支持する証拠も幾つか見られるものの、適用順序を指定しなければならない証拠も厳に存在する。適用環境の記述を工夫することで適用順序の指定を回避できるのか。

目的の項（§2）で断ったとおり、本稿はアッカド語動詞体系派生の骨組みを扱ったに過ぎず、今回漏らした項目が幾らか残されている。一つは、命令形であり、一つは、弱動詞または「不規則動詞」と言われる形式である。アッカド語を含むセム語においては、「不規則」動詞とは言っても、音韻規則で派生可能な形式である。少なくとも、インド・ヨーロッパ諸語における不規則動詞ほどの個別的処理を要求する性質のもの、あるいは辞書で扱うべき類のものではない。今後は、それらの形式をも含めた活用形を派生できるシステムを工夫する必要がある。

補遺

本稿の活用形生成プログラムの試行結果を幾つか例示する。

```
c:¥>
```

上のようにプロンプトが表示されたら、バッチ・プログラム名に続いて、①語幹、②活用形、③人称接辞、④語幹母音、の順で入力する：

```
c:¥>paradigm  g  dur  m 3 s  o
```

これは、①G語幹・②現在形・③3人称単数男性形・④語幹母音交替動詞の活用形を要求しているのである。続いてリターン・キーを押すと、その結果が次のように行頭に人称接辞の見出しを伴って出力される。

```
c:¥>
m3s       iPaRRaS
```

ここで、項目③を入力する際に、特定の人称・性・数を指定せずにピリオドを入力すると、一組の人称接辞を含む活用表全体を表示することができる。例えば：

```
c:¥>paradigm  ntn  pret  .  o
```

と入力すると、次のような出力結果を得る。

```
c:¥>
m3s     ittaPRaS
f3s     tattaPRaS
m2s     tattaPRaS
f2s     tattaPRaSi:
c1s     attaPRaS
m3p     ittaPRaSu:
```

f3p ittaPRaSa:
c2p tattaPRaSa:
c1p nittaPRaS

さらに、別の入力例を示す。

c:¥>paradigm sta stat . o

その結果、次の活用表が出力される。

c:¥>
m3s $utaPRuS
f3s $utaPRuSat
m2s $utaPRuSa:ta
f2s $utaPRuSa:ti
c1s $utaPRuSa:ku
m3p $utaPRuSu:
f3p $utaPRuSa:
m2p $utaPRuSa:tunu
f2p $utaPRuSa:tina
c1p $utaPRuSa:nu

参照文献

Huehnengard, J., *A Grammar of Akkadian*, Atlanta, 1997
Miller, D. B. and R. M. Shipp, *An Akkadian Handbook*, Winnona Lake, 1996
Soden, W. von, *Grundriss der akkadischen Grammatik*, Roma, 1969[2)]

Abstract

In this paper I tried to make explicit the phonological rules which generate principal paradigms of Akkadian strong verbs, including almost all the derived stems, modelled on the root morpheme √prs. This aim is simply realized and tested by a batch file and a SED script file with two additional files which contain affixes of person, gender and number. These small files will facilitate the learning of complicated Akkadian verbal paradigms, and make the language more accessible.